applied sciences

MDPI

Article

Differential Evolution: A Survey and Analysis

Tarik Eltaeib * and Ausif Mahmood

Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT 06614, USA; mahmood@bridgeport.edu

* Correspondence: teltaeib@bridgeport.edu; Tel.: +1-678-237-6229

Received: 26 July 2018; Accepted: 4 October 2018; Published: 16 October 2018

Abstract: Differential evolution (DE) has been extensively used in optimization studies since its development in 1995 because of its reputation as an effective global optimizer. DE is a population-based metaheuristic technique that develops numerical vectors to solve optimization problems. DE strategies have a significant impact on DE performance and play a vital role in achieving stochastic global optimization. However, DE is highly dependent on the control parameters involved. In practice, the fine-tuning of these parameters is not always easy. Here, we discuss the improvements and developments that have been made to DE algorithms. In particular, we present a state-of-the-art survey of the literature on DE and its recent advances, such as the development of adaptive, self-adaptive and hybrid techniques.

Keywords: differential evolution; optimization; stochastic

1. Introduction

Optimization algorithms are important approaches for resolving difficult optimization problems [1]. Optimization is defined as the procedure of discovery that provides the minimum or maximum value of a function $f(x)$ [2,3]. There are many reasons that make these problems difficult to solve. First, we cannot perform a comprehensive search if the problem domain space is too large. Second, the evaluation function is noisy or varies with time, generating a series of solutions instead of a single solution. Third, sometimes the constraints prevent arriving at a possible solution such that the optimization approach is the only solution [4]. The mathematical representation for the optimization involves finding an $X^* = [x_1^*, x_2^*, x_N^*] \in D^N = D_1 \cap D_2 \cap \ldots \cap D_N$, where

$$f_i^{min}(X^*) \leq f_i^{min}(X) \forall \mathrm{x} = [x_1\, x_2 \ldots x_N] \in D^N, 1 \leq i \leq N_{f^{min}}$$

$$f_i^{max}(X^*) \geq f_i^{max}(X) \forall \mathrm{x} = [x_1\, x_2 \ldots x_N] \in D^N, 1 \leq i \leq N_{f^{max}}$$

where X^* is the optimal solution in the N-dimensional search space D^N, N is the dimension of the optimization (the number of optimization parameters) and $f_i^{\min}$ and $f_i^{\max}$ are the objective functions.

Differential evolution (DE) is a stochastic algorithm for solving numerical continuous optimization problems. Since its inception, the DE algorithm has become a powerful global optimizer. Developed by Kenneth Price in 1994, DE is a promising optimization algorithm that converges to the real optimum without using significant amounts of resources. Furthermore, its performance was validated in the evolutionary domain by the IEEE Conference on Evolutionary Computation (CEC) in 1996 [5].

More recently, different versions of DE have secured the top ranks in many competitions between evolutionary algorithms (EAs) by the IEEE CEC conference series (http://www.ntu.edu.sg/home/epnsugan/index_files/cec-benchmarking.htm). However, an impressive number of different DE algorithms have been introduced by the research community over the past decades. To understand all the improvements that have been made to DE, a theoretical study and relative analysis of different

enhancements are presented in this paper. Because various DE algorithms involving different techniques are comprehensively discussed, the main motivation behind this survey is to deepen understanding of the characteristics of different DE strategies, with the goal of benefiting from the various approaches. In fact, understanding how to combine these DEs harmoniously and their underlying concepts could be crucial to attaining effective designs or improving the performance of DE algorithms in particular or any optimization algorithms in general. Moreover, the literature shows that no single algorithm has been demonstrated to be effective for various applications. Thus, this study may provide a roadmap through which developers may gain a full understanding of this field.

DE algorithms are different from EA algorithms that shape offspring by mixing solutions with a difference factor rate of selected individual vectors and they are an alternative to recombining individuals through a probabilistic scheme. In fact, the differential mutation strategy is the main component that distinguishes DE from other population algorithms. Applying the mutation to all candidates defines an exploration rule based on other candidate solutions. Therefore, the mutation strategy enhances a population's capability for discovering new promising offspring based on the current distribution of solutions within the domain space [6]. Ideally, the performance of DE is based on two major components: the chosen strategy and the control parameters. However, the strategy underlying DE consists of mutation, crossover and selection operators, which are utilized at each generation to determine the global optimum. The control parameter components consist of the population size NP, scaling factor F and the crossover rate Cr [7,8].

Despite the potential of DE, it is obvious to the research community that some adjustments to classic DE are essential to significantly enhance its performance, especially in addressing high-dimension problems. Stagnation, premature convergence and sensitivity to control parameters are the main issues that influence the performance of DE [9]. Stagnation occurs when the population cannot converge to a suboptimal solution although the diversity of the population remains high [10]. In other words, the population does not improve over a period of iterations, and the algorithm is not capable of finding a new search domain [11]. There are many causes of stagnation, including control parameters, which become inefficient for a specific problem in the decision space [9,10]. Many studies have proposed a variety of ways to improve the current DE algorithm through modifications [12,13], including the use of differential mutations with perturbations, mutations with selection pressure, and operator adaption techniques [14–17]. Qing conducted an extensive study on differential evolution [18] and observed that the performance of differential evolution and the quality of the results were based on the type of technique used, either binomial or exponential [16,19].

This article provides a comprehensive survey of the different types of state-of-the-art differential evolution algorithms available as global numerical optimizations over continuous search spaces. Thus, this study provides a foundation for researchers who are interested in optimization in general or who care about recent developments in DE. The growing research area is divided into adaptive, self-adaptive, and hybridization strategies. This comprehensive study sheds light on most improvements and developments pertaining to different types of DE families, including primary concepts and a variety of DE formats.

This article is organized as follows. Section 2 briefly describes classic differential evolution. Section 3 analyzes the parameter strategies and their effects on DE performance and surveys different DE approaches. Section 4 presents the conclusions drawn from our study.

2. Classic Differential Evolution

If we are seeking the optimum for X^* which is demonstrated by vector X_i^*, $i = 1, \ldots$ D, $X \in \mathbb{R}^D$, within boundary constraints $L \leq X \leq H$. Differential evolution (DE) is population-based, where the initial population is $P_{i,j} \in \mathbb{R}^{[D \times NP]}$ with random initialization $H \leq P_{i,j} \leq L$. Initialization of the population is an important step that assumes that there is no previous information about the optimum

solution. Therefore, the population is initialized within only boundary constraints, the upper bound (H) and lower bound (L), so the population can by initialized by the following:

$$P_{i,j} = L + (H - L) \cdot rand_{i,j}[0,1) \qquad i = 1, 2, ..., D; j = 1, 2, ..., NP$$

After the initialization phase, the evolution involves the three processes of mutation, crossover, and selection. The classic differential evolution strategy consists of three random vectors v_1, v_2 and v_3 that are selected from the population (Equation (1)). Randomly select three individuals from the population $v_{1,2,3} \in [1, ..., NP]$, where $v_1 \neq v_2 \neq v_3$

$$v_1 = (int)\ (rand() * NP);$$

$$v_2 = (int)\ (rand() * NP);$$

while $(v_2 = v_1)\ v_2 = (int)\ (rand() * NP)$;

$$v_3 = (int)\ (rand() * NP);$$

while $(v_2 = v_1\ ||\ (v_2 = v_3)\ v_3 = (int)\ (rand() * NP))\ v_2 = (int)\ (rand() * NP)$.

The mutation operation recombines to construct the mutation vector v_y shown in Figure 1. The associated equation is

$$v_y = v_3 + F * (v_1 - v_2) \tag{1}$$

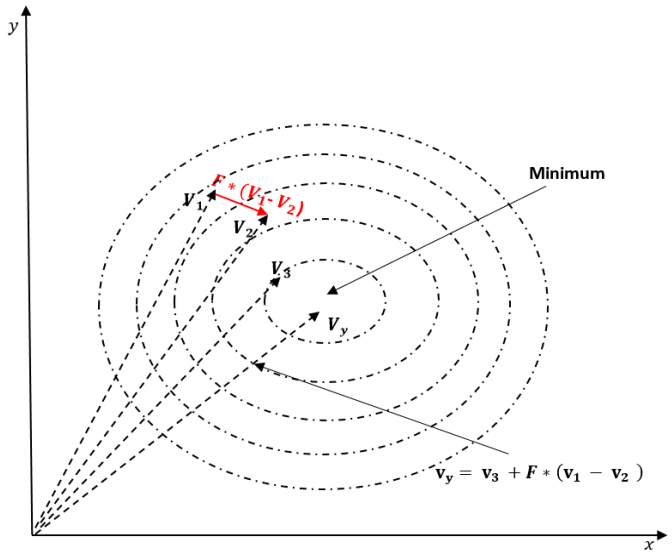

Figure 1. Random vectors selected in the mutation strategy (classic differential evolution (DE)).

The mutation process is the main distinctive component of DE and is considered the strategy by which DE is carried out. There are different types of mutation strategies, each one distinguished with an abbreviation based on the classic mutation strategy described by Equation (1), i.e., DE/rand/1/bin, where DE represents differential evolution and "rand" represents random, which indicates that the vectors are selected randomly. The number one indicates the number of difference pairs; in this strategy, it is one pair $(v_1 - v_2)$. The last term represents the type of crossover used. This term could be "exp", for exponential, or "bin", for binomial [20]. Then, to complement the previous step (mutation strategy), DE also applies uniform crossover to construct trial vectors v_{Trail}, which are out of parameter values that have been copied from two different vectors. In particular, DE selected a random vector from the population, indicated as v_x , which must be different to v_1, v_2, and v_3, and then it crosses with a mutant vector v_y; the binomial crossover is generated as follows:

$$v_{Trail}[i,j] = \begin{cases} v_y[i,j] \; if \; rand(0,1) < Cr \\ v_x[i,j] \; otherwise \end{cases} \quad (2)$$

The crossover probability, $Cr \in (0, 1)$, is a pre-defined rate that specifies the fraction of the parameter that is transferred from the mutant. Thus, it is used to control which sources participate in a given parameter. The uniform crossover rate is compared with uniform random values formed from rand (0, 1); if the random value is smaller than or equal to *Cr* then the trial parameter is copied from the mutant vector v_y, otherwise the parameter is inherited from v_x.

The next operation is selection, in which the trail vector v_{Trail} competes with the target vector v_x. If this trail vector v_{Trail} is equal or less than v_x it changes the target vector v_x in the next generation otherwise v_x is not changed in the population,

$$v_x = \begin{cases} v_{Trail} \; if \; f(v_{Trail}) \leq f(v_x) \\ v_x \; otherwise \end{cases} \quad (3)$$

where $f(x)$ is the objective function. If the new trail vector $f(v_{Trail})$ is less than or equal to the target vector $f(v_x)$, it replaces the target vector. Otherwise, the population maintains the target vector value. Therefore, the different DE phases prevent the population from ever deteriorating; the population either remains the same or improves. Furthermore, continued refining of the population is achieved by the trial vector, although the fitness of the trial vector is the same as that of the current vector. This factor is crucial in DE because it provides the algorithm the ability to move through the landscape using a variety of generations [21]. The termination condition can be either a preset maximum number of generations or a pre-specified target of the objective function value [22].

2.1. Differential Evolution Strategies

The various equations underpinning DE possess certain aspects in common when applied to continuous optimization. All consist of an original point, sometimes referred to as the base point. The original algorithm carries out the search operation such that it finds the optimum as soon as possible. We can generalize the DE formula to the form $\alpha = \beta + F \times \delta$, where β represents the base vectors and δ the difference between vectors. Thus, the main goal of all DE equations is to provide the optimal direction based on the differential β and base vector δ (Figure 2).

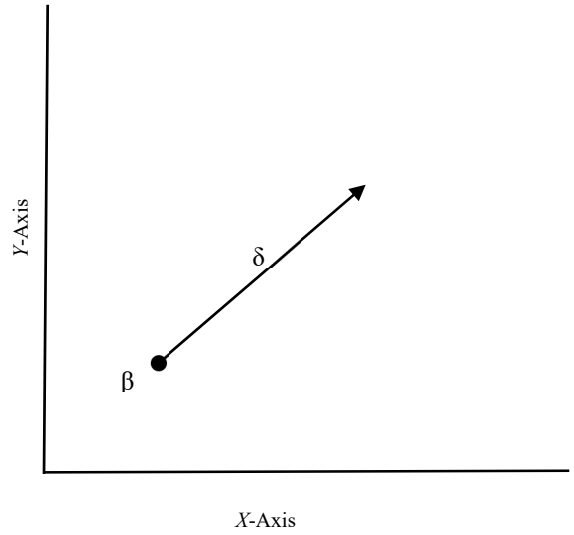

Figure 2. The differential β and base vector δ provide the optimal direction.

Establishing β and δ is crucial to creating an efficient strategy that can be applied to the chosen individuals from the population. However, all possible combinations of β and δ can be classified into

the following strategies: local, random, directed, and hybrid. In random strategies, abbreviated as "Rand", all individuals are formed randomly, and there is no prior information about the objective function. In directed strategies, abbreviated as "DIR", a suitable value for the base vector is chosen according to the objective function to ensure a suitable direction. Hybrid strategies include the combination of "Rand" and "DIR", labeled RAND/DIR. In another approach, the best overall vector is used, not only the best among the selected individuals; this approach is referred to as the "BEST". Combining the "Rand" and "BEST" yields the hybrid RAND/BEST strategy. In addition, the combination of more than two approaches, e.g., RAND/BEST/DIR, can yield favorable results by exploiting the advantages of each approach.

Table 1 shows that all DE strategies employed are formed based on the DE/rand/x variation, which applies pairs of difference vectors:

$$u_i = x_{r_1} + F_1(x_{r_1} - x_{r_1}) + \cdots\cdots\cdots + F_k(x_{r_{2k}} - x_{r_{2k}})$$

whereas the scaling factors are frequently presumed to be the same $F_1 = F_2 = \ldots = F_k = F$. Substituting an arbitrary base vector $\mathbf{x}_1$ as $\mathrm{v}_{\mathrm{best}}$, "the best vector" from the population, provides a different DE approach, indicated as DE/best/1:

$$\mathrm{v}_{\mathrm{y}} = \mathrm{v}_{\mathrm{best}} + F * (\mathrm{v}_1 - \mathrm{v}_2)$$

Most mutation strategies can be formed by a general formula based on the sum of k scaled difference vectors and a weighted average among the best vector and arbitrary ones:

$$\mathrm{v}_{\mathrm{y}} = \lambda \mathrm{v}_{\mathrm{best}} + (1-\lambda)\mathrm{v}_{\mathrm{x}} + \sum F_i * (\mathrm{v}_{r2_i} - \mathrm{v}_{r2_{i+1}})$$

Table 1. The differentiation operation can be carried out using many mutation strategies.

Strategy	Formulation
1. DE/best/1/exp	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} - V_{(2,G)}\right]$
2. DE/rand-to-best/1/exp	$V_{(G+1)} = V_{(i,G)} + \lambda\cdot\left[V_{(1,G)} - V_{(2,G)}\right] + F\cdot\left[V_{(1,G)} - V_{(2,G)}\right]$
3. DE/best/2/exp	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
4. DE/rand/2/exp	$V_{(G+1)} = V_{(5,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
5. DE/best/1/bin	$V_{(G+1)} = V_{(best,G)} + F\cdot\left[V_{(2,G)} - V_{(3,G)}\right]$
6. DE/rand/1/bin	$V_{(G+1)} = V_{(1,G)} + F\cdot\left[V_{(2,G)} - V_{(3,G)}\right]$
7. DE/rand-to-best/1/bin	$V_{(G+1)} = V_{(i,G)} + \lambda\cdot\left[V_{(best,G)} - V_{(i,G)}\right] + F\cdot\left[V_{(1,G)} - V_{(2,G)}\right]$
8. DE/best/2/bin	$V_{(G+1)} = V_{(best,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$
9. DE/rand/2/bin	$V_{(G+1)} = V_{(5,G)} + F\left[V_{(1,G)} + V_{(2,G)} - V_{(3,G)} - V_{(4,G)}\right]$

One aspect common to all the mutation strategy methods is the base vector, which controls the search direction. The difference vector provides a mutation rate term, such as a self-adaptive term, that is added to an arbitrary or guided base vector to construct a trial individual. Over generations, the individuals of a population reside in increasingly better positions and reform themselves. The various combinations of these vectors can be categorized into four groups based on information pertaining to the values gathered from the objective function: random, directed, local and hybrid.

The RAND approach consists of strategies in which the trial individual is produced without knowledge of the value of the objective function. Similarly, the RAND/DIR approach includes strategies that use the values of the objective function to determine a promising direction. Likewise, the RAND/BEST approach applies the best individual approach to proceed with a trial. Additionally, the RAND/BEST/DIR approach combines the last two groups into one that includes all of their

collective benefits. However, a suitable direction is obtained by using the best individual to decrease the search space and exploration time [23,24]. Thus, the "dir" and "dir-best" strategies, which use objective function values to generate trial individuals, can produce an exploitation function. In fact, the random selection of parents for a trial enhances exploration capabilities [25–27]. Thus, the locations of individuals carry information about the fitness landscape. Therefore, an effective mutation strategy that leads to uniform random vectors represents the entire search space well.

2.2. Initialization

DE is a population-based optimization technique that begins with the problem solution by selecting the objective function at a random initial population. Predefined parameter bounds describe the area from which the number of population (N_p) vectors in this initial population is chosen within both the upper bound "b_U" and the lower bound "b_L", where the subscripts L and U indicate lower and upper, respectively. The following equation is used to develop a random number generator for all vectors from within the predefined upper and lower bounds. The random function Random (0, 1) produces a uniform random number within the range (0, 1).

$$x_i = Random(0,1)\cdot (b_U - b_L) + b_L$$

2.3. Crossover

To balance the differential mutation search strategy, DE also applies uniform crossover to construct trial vectors. A trial vector is constructed from values that have been copied from two diverse vectors. In particular, DE crosses each vector as follows:

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} \text{If}(rand_j(0,1) \le Cr \text{ or } j = j_{rand}) \\ x_{j,i,g} \end{cases}$$

The crossover probability, $C_r \in [0, 1]$, is predefined in the classic version of DE, and the fraction value of the Cr control is cloned from the mutant vector. C_r is compared with a random number randj (0, 1). If the random number is less than or equal to C_r, the trial parameter is inherited from the mutant $v_{j,i,g}$, otherwise, the parameter is cloned from the vector $x_{j,i,g}$.

2.4. Selection

In this stage, we determine when the trial vector $u_{i,g}$ has an objective function value that is less than or equal to that of its target vector $x_{i,g}$. DE swaps the target vector in the next iteration, otherwise the target retains its place in the population. This process is carried out by comparing each trial vector with the target vector from which the parameters are cloned. After the population is updated, mutation, recombination and selection are repeated until the optimum value is found or after a predefined stop criterion is reached, such as a certain number of iterations.

$$x_{i,g+1} = \begin{cases} u_{i,g} \text{If}(f(u_{i,g}) \le f(x_{i,g}) \\ x_{i,g} \end{cases}$$

2.5. Differential Evolution Paused-Code and Flowchart

The flow chart of classic differential evolution is shown in Figure 3, and a pseudo-code for classic differential evolution is given in Algorithm 1, which provides a pseudo-code of the DE algorithm for minimizing a cost function, specifically, a DE/rand/-1/bin strategy.

Algorithm 1. Pseudocode for classic Differential Evolution.

Input: Population size '*NP*', Problem Size 'D', Mutation Rate '*F*', Crossover Rate '*Cr*'; Stope_Criteria {Number of Generation, Target}, Upper Bound 'U', Lower Bound 'L'
Output: Best_Vector

```
Population = Initialize Population (NP, D, U, L);
While (Stope_Criteria ≠ True) do
    Best_Vector = EvaluatePopulation (Population);
     v_x = Select_Random_Vector (Population);
     Index = FindIndexOfVector (v_x); //specify row number of a vector
     Select_Random_Vector (Population,v_1, v_2, v_3) where v_1 ≠ v_2 ≠ v_3 ≠ v_x
   Vy = v_1, + F(v_2 − v_3)
          For (i = 0; i++; i < D − 1)//Loop for starting Crossover operation
               if (rand_j [0, 1) < CR)Then
               u[i] = v_x [i]
               Else u[i] = v_y [i]
          End For Loop//end crossover operation
     If (CostFunctionOfVector(u) ≤ CostFunctionOfVector (v_x)) Then
     UpdatePopulation (u, Index, Population);
  End; //While loop
Retune Best_Vector;
```

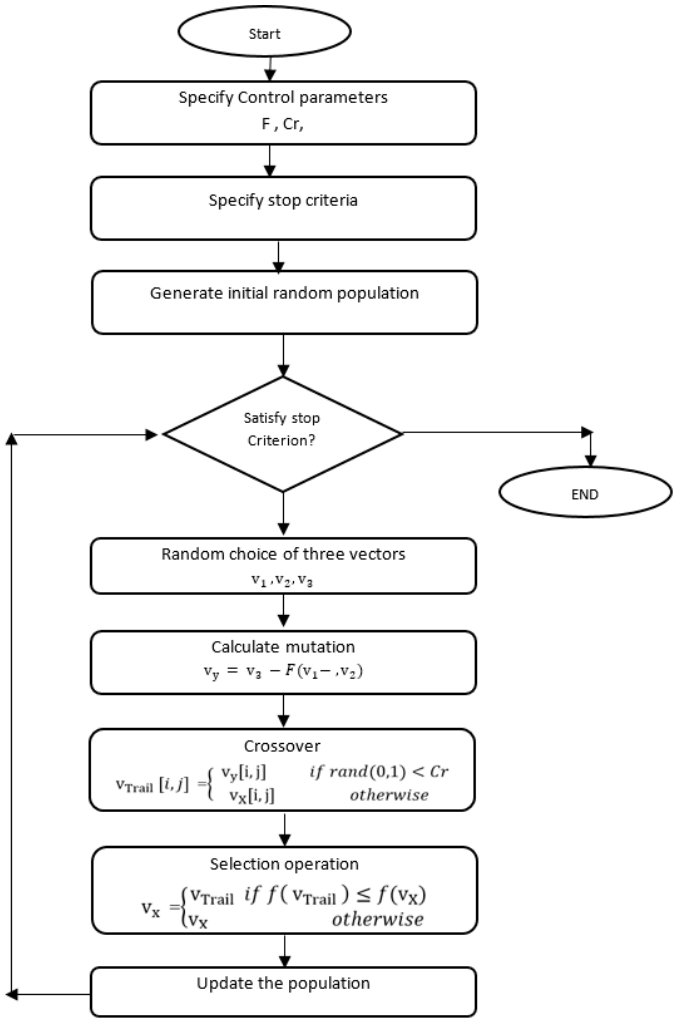

Figure 3. Pseudo-code of classic differential evolution.

2.6. DE Applications

Due to the rapid rise of DE as a modest and strong optimizer, developers have applied the technique in a wide range of domains and fields of technology (http://www1.icsi.berkeley.edu/~storn/code.html). Yalcin proposed a new method for the 3D tracking of license plates from video using a DE algorithm, which could be fine-tuned according to the license plate boundaries [28]. A color image quantization application using DE was proposed by Qinghua and Hu. The main objective of image processing techniques during the color image quantization phase, is to decrease the number of colors in an image with a low amount of deformation. DE can be used to adjust colormaps and find the optimal candidate colormap [29]. With respect to the bidding market, Alvaro et al. applied DE in developing a competitive electricity market application that finds the optimal bids based on daily bidding activity [30]. Sickel et al. used DE in developing a power plant control application for a reference governor to produce an optimal group of points for controlling a power plant that was produced by [31]. Wang et al. proposed a flexible QoS multicast routing algorithm for the next-generation Internet that improves the quality of service (QoS) of multicasts to manage the increasing demand of network resources [32]. With respect to the electric power systems industry, Ela et al. applied DE to determine the optimal power flow [33]. Goswami et al. proposed a DE application for model-based well log-data inversion to discover features of earth formations based on the dimensions of physical phenomena [34]. Another application applies network system reconfiguration for distributing systems. The network reconfiguration application proposed by Tzong and Lee involves the application of Improved Mixed-Integer Hybrid Differential Evolution [35]. Another DE application developed by Boughari et al. sets suitable controllers for aircraft stability and control augmentation systems [36].

3. Parameter Control

As suggested in the previous section, the DE algorithm is a simple and effective optimization algorithm for problems from the real world when its control parameters are properly set [8,37,38]. In this section, we review the most current approaches for improving DE. First, the DE algorithm applies certain control parameters to the system implementation. The accomplishment of DE is influenced by the value of parameters, such as the crossover and mutation rate. Although some studies have recommended certain values for these parameters, their effect on performance is complex and their exact values are unclear. In particular, there is a wide variety of different recommended values that are appropriate for different problems [39–41].

The mutation rate "F", crossover rate "C_r" and population dimension "Np" maintain balance between exploration and exploitation [6]. Exploration is associated with finding new solutions, and exploitation is associated with searching for new, suitable solutions; the two processes are linked in the evolutionary search [42,43]. Therefore, the mutation and crossover rates influence the convergence rate and the effectiveness of the search space [44].

However, specifying suitable values for these rates is not easy [45]. Three types of strategies are used to set these parameter controls: deterministic parameter control (sometimes called random), self-adaptive parameter control and adaptive parameter control [21,46–48]. Adaptive and self-adaptive parameter control [16–25] have recently been proposed to dynamically alter the control parameters without requiring the user's prior knowledge or information about the problem behaviors throughout the search process [49–53]. In the following sections, the self-adaptive parameter, the adaptive parameter, and hybrid control strategies are discussed.

3.1. Deterministic Parameter Control

The parameters are altered using a deterministic rule regardless of the feedback from the evolutionary search, with jitter and dither being two operators that are used in this technique. Dither scales the distance of the vector differentials as the same factor, F_i, is applied to all the elements

of a subtracted vector. Jitter multiplies each vector element of the subtracted vector by a different scale factor, F_j. The rotation creates jitter using an essentially different procedure than the classic DE's constant mutation with F. However, this approach shows robustness for non-deceiving objective functions [3]. Nonetheless, applied fixed values for each iteration, and F was created for each individual within the range (0.4, 1), whereas the interval (0.5, 0.7) was selected for Cr [54,55].

Another approach is the composite DE (CoDE) algorithm proposed by Wang et al. In CoDE, a trial vector is selected from a set of groups produced by utilizing diverse DE strategies [56]. The main objective is to arbitrarily merge many trial vector strategies with different parameters at each iteration to construct new trial vectors. These combinations help to successfully solve many problems. Wang et al. used a group of trial vector strategies and a group of control parameter (almost three) to create strategy and parameter candidate pools. The selected strategies are DE/rand/1/bin, DE/rand/2/bin and DE/current-to- rand/1, and the three pair common choices for the control parameter settings were (F = 1.0, Cr = 0.1), (F = 1.0; Cr = 0.9), and (F = 0.8; Cr = 0.2). In each generation, three different strategies are applied, which randomly pick any of the control parameter values. Then, the trial vector is designated the candidate with the better value of fitness. The parameters are chosen based on whether they are frequently implemented with many DEs, and their performances are evaluated. The three pairs of parameter settings that provide diverse effects produce new improved candidates. Furthermore, the different values of the control parameters maintain different levels of search performance.

3.2. Adaptive Parameter Control

The adaptive technique has been applied with classic DE/rand/1/bin. While the performance is relatively favorable, the technique still suffers from convergence rate issues [50,51]. If very well designed, self-adaptive and adaptive parameter controls can enhance the robustness and the convergence rate by automatically adapting to the parameters. Approaches other than using the best explored solution, use minor resolution in previous generations and their variation with the present population as a good area for finding the optimum. Adapting the parameters is a method called the adaptive DE algorithm (ADE), which applies an evaluation from feedback of the F relay on an additional parameter (γ) that it is necessary to be adjusted [57,58]. However, the self-adaptive parameter controls the value assignments and adjusts them dynamically. A parameter is altered dynamically through processing according to pre-defined rules using adaptive control, self-adaptive control or a combination thereof [45,59].

The main purpose of adaptive DE is to help exploit and explore relationships that avoid premature convergence problems and to optimize the final results. In general, there are many techniques for hybridizing a conventional evolutionary algorithm to solve optimization problems. The initial population of DE is formed by problem-specific heuristics. Then, other solutions obtained using another EA might be enhanced with a local search. This type of combination is called a memetic algorithm [21,59]. The benefits of this hybridization lead to various operators that might exploit problem knowledge, such as merging more promising individuals to be inherited. Furthermore, mutation operations may be biased to contain solutions of promising individuals with higher probabilities than those of others.

3.2.1. Differential Evolution with Self-Adapting Populations (DESAP)

Differential evolution with self-adapting populations (DESAP) dynamically adjusts the crossover and mutation parameters δ, η and the population size π [17]. Each individual i is connected to its control δ_i, η_i, and π_i. δ and π have similar meanings to NP and Cr, correspondingly. The mutation factor F is retained as static, and η denotes the probability of implementing an extra mutation operation by using normally distributed. The main technique of DESAP is unlike that of the traditional DE/rand/1/bin algorithm [16]. Parameters are adapted by developing them over the mutation and crossover processes, as the procedures are applied to each x_i. The updated values of the parameters continue with u_i if $f(u_i)f(x_i)$. However, DESAP still requires further development to produce better performances. In

fact, despite its simplicity, DESAP performed better than DE in one of De Jong's five exam problems, whereas the other solutions are almost identical. DESAP represents an opportunity to reduce the control parameters further by updating the size of the population, as is done with the additional parameters.

3.2.2. Fuzzy Adaptive Differential Evolution (FADE)

Fuzzy adaptive differential evolution (FADE), presented by Lampinen and Liu [19], is a different type of DE algorithm that applies fuzzy logic controllers to adjust the controller parameters F_i and Cr_i for the crossover and mutation operations. Similarly to DESAP, the size of the population is presumed to be adjusted and is static during the evolution procedure [20]. The fuzzy-logic control method has been verified on a group of 10 functions as a benchmark and displays better solutions than those of classic DE for high-dimensional problems.

3.2.3. Self-Adaptive Differential Evolution (SaDE)

Self-adaptive differential evolution (SaDE) is simultaneously applied to a pair of mutation techniques "DE/rand/1" and "DE/current-to-best/2" [52]. The adaptation technique of the parameter consists of two chunks: the probability of the adaptation p_i, where i = (1, 2), and the DE parameters F_i and Cr_i. The probability of producing a mutation vector based on the two strategies approaches is 0.5 and this is updated every 50 iterations using the following method:

$$p_1 = \frac{ns_1\ (ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)}$$

$$p_2 = 1 - p_1$$

where ns_{i} and nf_i are the numbers of offspring vectors constructed by the ith i = (1, 2) strategy that was a success or failure in the selection process over the last 50 generations. It is assumed that this adaptation process can progressively develop the most appropriate mutation strategy at diverse learning phases for a given problem. The mutation factors F_i are autonomously created at each iteration based on a normal distribution, NR with a mean of 0.5 and a standard deviation of 0.3,

$$F_i = NR(0.5,\ 0.3)$$

The crossover rates Cr_i are autonomously formed based on a normal distribution with a mean of Cr_m and a standard deviation of 0.1. The mean Cr_m approaches 0.5, is changed every 25 iterations and is set to be the mean of the effective Cr over the previous 25 generations.

$$Cr_i = NR(Cr_m,\ 0.1)$$

$$Cr_m = \frac{1}{k}\sum_{k=1}^{k} Cr_{suc}(k)$$

where K is the counters of effective Cr values and $Cr_{suc}(k)$ indicates the k_{th} value.

To accelerate the convergence, a local search technique (Quasi–Newton method) is applied to respectable individuals after 200 generations. SaDE has been further developed by applying five mutation strategies to resolve a group of constrained problems [60].

3.2.4. Self-Adaptive NSDE (SaNSDE)

Neighborhood search differential evolution (NSDE) is similar to classic DE except that Equation (1) is replaced with

$$\mathrm{v_y} = \mathrm{v_3} + \begin{cases} d_i * N(0.5,\ 0.5)\ if\ u(0.1)0.5 \\ d_i * \delta\ otherwise \end{cases}$$

where $d_i = v_1 - v_2$ is the differential deviation, N (0.5, 0.5) means a Gaussian random number with an average of 0.5 and a standard deviation of 0.5 and δ indicates a Cauchy random variable with a rate parameter of t = 1.

Self-adaptive DE (SaDE) [8] was developed to resolve the issues related to control parameters and learning technique. In SaDE, two DE learning strategies are chosen according to their performance. The most appropriate learning technique and parameter values are increasingly self-adapted according to the learning experience gained during evolution [61].

SaNSDE is an adaptive differential evolution algorithm that produces mutation vectors in a manner similar to SaDE [61]. However, the difference is that the mutation factors are established based on a normal distribution or a Cauchy distribution:

$$f = \begin{cases} Normal\ distribution(0.5,\ 0.3),\ if\ rand > fp \\ Cauchy\ distribution(0.0,\ 1.0),\ \text{otherwise} \end{cases}$$

where the normal distribution (μ,σ^2) indicates a random value of mean μ and variance σ^2 and a Cauchydistribution (μ, δ) indicates a random value with scale parameters μ and δ. The probability fp of the spread over is adapted as follows:

$$p_1 = \frac{ns_1(ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)}$$

The crossover rate adaptation is similar to the method used in SaDE, but the factor Cr_m is changed as a biased average of the successful values Cr_{suc} every 25 iterations.

$$Cr_m = \frac{1}{k}\sum_{k=1}^{k} W_k Cr_{suc}(k)\ W_k = \frac{\Delta_K}{\sum_{k=1}^{k} \Delta_K}$$

where the weight is calculated with a positive improvement $\Delta = f(x) - f(u)$ in the selection related to each successful crossover rate CRsuc (k).

3.2.5. Self-Adapting Parameter Setting in Differential Evolution (jDE)

jDE is another adaptive DE algorithm that is similar to the classic DE/rand/1/bin algorithm. jDE improves the population size throughout the optimization process based on the improved parameters and thus generates vectors that are more likely to survive. However, the mechanism of jDE involves adapting the parameters F_i and CR_i associated with each individual. At the beginning of the process, the parameter values are F_i = 0.5 and CR_i = 0.9 for each individual. However, F_i and CR_m are updated from the effective records; thus, jDE produces new values within the probabilities $\tau_1 = \tau_2$, which are used to alter the control parameters. The updated values for F_i and CR_i are then obtained using uniform distributions over (0.1, 1) and (0, 1), respectively. That is,

$$F_{i,\,g+1} = \begin{cases} 0.1 + 0.9 * random1,\ if\ random2 < \tau_1 \\ F_{i,\,g}\ otherwise \end{cases}$$

$$CR_{i,\,g+1} = \begin{cases} random3,\ if\ random4 < \tau_2 \\ CR_{i,\,g}\ otherwise \end{cases}$$

where *random* j = 1, 2, 3, 4 is the uniform random function $\in$[0, 1]. The updated parameters are implemented in the mutation and crossover processes to produce new, consistent vectors. This mechanism updates the prior parameter with a new one only if the new vectors pass the selection phase. However, jDE yields improved results with the classic DE/rand/1/ bin strategy.

3.2.6. Adaptive DE Algorithm (ADE)

Hu and Yan proposed another adaptive DE algorithm. They modified the parameters F and Cr to each iteration using the current generation and the fitness [62]. They tried to find the optimal value for the parameters F and Cr to find a balance between reliability and efficiency. The mutation and crossover operations are calculated for each generation. Thus, for each parent x_i^g of generation g, the offspring x'^{g+1}_i is constructed as follows: calculate the Gth mutation $F(G)$ and crossover $CR(G)$ as,

$$\begin{cases} F(G) = F_0 * (\frac{\exp(I_G)-\exp(-I_G)}{40} + 1) \\ CR(G) = CR_0 * (\frac{\exp(I_G)-\exp(-I_G)}{40} + 1) \end{cases}$$
$$I_G = 3 - 6 * (\frac{G}{G_M})$$

3.2.7. Modified DE (MDE)

MDE uses only one array, which is updated when a better solution is found. Therefore, continuously updating the one array improves the convergence speed, leading to fewer evaluation procedures than those associated with classical DE [63]. In MDE, and by applied distribution of Laplace "F" is arbitrarily adjusted [63]. The Laplace distribution is analogous to the (NP) normal distribution [64]. Moreover, the Laplace distribution is longer and skewed, allowing for inference so that it can control more efficiently, thus avoiding premature convergence. Experimental results demonstrate that modified DE with a Laplace distribution (MDE) offers enhanced performance compared with the classical DE approach [65].

3.2.8. Modified DE with P-Best Crossover (MDE_pBX)

MDE_pBX involves F and Cr values that are produced using a Cauchy distribution using a position parameter, and then adapted on the power average of entirely F/Cr ratios producing effective offspring [66]. The mutation strategy used in this algorithm scheme (DE/current-to-best/1) can be expressed as follows:

$$V_{i,g} = X_{i,g} + F_i\,(X_{gr_{best},G} - X_{i,G} + X_{r_1^i,G} - X_{r_2^i,G})$$

where $X_{gr_{best},G}$ is the finest of the q% vectors arbitrarily selected from the existing generation, whereas $X_{r_1^i,G}$ and $X_{r_2^i,G}$ are two distinctive vectors chosen randomly from the current population and are not equal to $X_{gr_{best},G}$ or the target.

In the p-best crossover process, each different random vector is chosen from the p best-ranking vectors in the present population [67]. Then, a standard crossover is executed as per (5) between the vector and the arbitrarily chosen one from the p-top vector to produce the trial vector with an identical index. The variable p is linearly made smaller with the following generations as shown:

$$p = ceil\left[\frac{Np}{2} \cdot (1 - \frac{G-1}{G_{max}})\right]$$

where G is the present generation value, G_{max} is the most extreme number of generations and Np is the population number. The parameter adaption mechanism F_i is independently calculated as,

$$Fi = \text{Cauchy Distribution}\,(F_m, 0.1)$$

$$F_m = w_f \cdot fm + (1 - w_f) \cdot mean_{Pow}(F_{success})$$

where F_m initialized with 0.5

$$w_f = 0.8 + 0.2 * \text{ rand}\,(0,1)$$

$$mean_{Pow}(F_{success}) = \sum_{x \in F_{success}} \left(x^n \Big/ (|F_{success}|)^{\frac{1}{2}}\right)$$

where n = 1.5 and $|F_{success}|$ is the set of cardinality.

The crossover probability adaptation Cr of each individual vector is independently created as

$$Cri = \text{Gaussian Distribution}\ (Cr_m, 0.1)$$

$$Cr_m = w_{Cr} \cdot Cr_m + (1 - w_{Cr}) \cdot mean_{Pow}(Cr_{success}) w_{cr} = 0.8 + 0.2 * \text{ rand}\ (0,1)$$

$$mean_{Pow}(Cr_{success}) = \sum_{x \in Cr_{success}} \left(x^n \Big/ (|Cr_{success}|)^{\frac{1}{2}} \right)$$

where n = 1.5 and $|Cr_{success}|$ is the set of cardinality.

3.2.9. DE with Self-Adaptive Mutation and Crossover (DESAMC)

DE with self-adaptive mutation and crossover (DESAMC) is a new version of DE [68,69]. In this approach, F is adapted using an affection index (Af_i), calculated using information about fitness. A minor Af_i shows that each one is far away from the best global vector (best solution); consequently, a robust global exploration is essential. The formula of adaptation is as follows:

$$F_i(g) = \frac{1}{1 + \tanh(2Af_i(g))}$$

where tanh indicates the hyperbolic tangent function

$$\tanh(z) = \frac{\exp(2z) - 1}{\exp(2z) + 1}$$

where the crossover is

$$CR(t) = CR^- + (1 - \frac{t}{t_{max}})^{\frac{t}{t_{max}}} (CR^+ - CR^-)$$

where t is the present generation, t_{max} is the greatest number of generations and CR^+ and CR^- are the maximum and minimum values of CR, respectively.

3.2.10. Adaptive Differential Evolution with Optional External Archive (JADE)

JADE is an alternative to adapting the parameters at each generation toward progressive self-adaptation, based on the success rate [29]. Qin and Suganthan [52] and Zhang and Sanderson [29] proposed the new mutation strategy (DE/current-to-pbest/1). Furthermore, they used new adaptive parameters, μ_{CR} and μ_F.

The crossover and selection operations are implemented as in the classic DE algorithm.

The greedy strategy involves a new mutation strategy called DE/current-to-pbest/1 (without archives) and assists the baseline JADE:

$$V_{i,g} = X_{i,g} + F_i\ (V_{best,\ g} - V_{1,g}) + F_i\ (V_{2,g} - V_{3,g})$$

$$V_{i,g} = X_{i,g} + F_i\ (V_{best,\ g} - V_{1,g}) + F_i\ (V_{2,g} - V'_{3,G})$$

where $V_{best,g}$ is the best solution that is randomly chosen as one of the best individuals from the current population [56]. Similarly, $V_{1,g}$, $V_{2,g}$ and $V_{3,g}$ are randomly selected from the current population. However, $V'_{3,G}$ is also randomly chosen from the union between $X_{i,g}$ and $V_{1,g}$.

$$V'_{3,G} = \text{randomly}\ (V_{1,g} \cup X_{i,g})$$

JADE is also applied to the archiving process. Initially, the archive is unfilled and is added to the parent solutions that fail in the selection process [70]. The purpose of the archive is to avoid calculation

overhead. Moreover, the archive has a limited size; thus, if the size of the archive grows beyond R, then a shrink operation is performed to reduce its size so that it does not exceed (α, NP).

The archive technique provides information about the direction required to improve the diversity of the population. In addition, arbitrary F values can help expand population diversity [71].

3.2.11. Adaptation of $\mu_{CR}\mu_F$

The adaptation technique used for JADE is applied to μ_{CR} and μ_F to produce the mutation rate F_i and the crossover rate CR_i related to each individual vector xi. JADE is implemented in each iteration i, and the crossover rate CR_i of each individual xi is individually formed based on a normal random distribution = Normal Distribution (μ_{CR}, 0.1), where the mean μ_{CR} is initially 0.5 and the standard deviation is 0.1, i.e.,

$$CR_i = \text{Normal Distribution } (\mu_{CR}, 0.1).$$

Then, S_{CR} is calculated, which represents the set of all effective crossover rates CR_i. Furthermore, the parameter μ_{CR} is updated in each iteration; this information is saved, and random information is deleted from the archive file to keep its size R. μ_{CR} is calculated as follows:

$$\mu_{CR} = (1 - c)\cdot\mu_{CR} + c\cdot mean(S_{CR})$$

Similarly, the mutation rate F_i is calculated using the Cauchy distribution (μ_F, 0.1), with the constraint that F_i = 1.

If $F_i \geq 1$ or $F_i \leq 0$ and μ_f is initialized as 0.5, then

$$Fi = \text{Cauchy Distribution } (\mu_f, 0.1)$$

where S_F indicates the set of all effective mutation rates F_i. Then, μ_f is updated as follows:

$$\mu_f = (1 - c)\cdot\mu_f + c\cdot mean_L(S_F)$$

where $mean_L$ indicates the Lehmer mean which is calculated as follows:

$$(S_F) = \frac{\sum_{F\in S_F} F^2}{\sum_{F\in S_F} F}$$

3.2.12. Differential Covariance Matrix Adaptation Evolutionary Algorithm (CMA-ES)

Saurav et al. proposed the Differential Covariance Matrix Adaptation Evolutionary Algorithm for real parameter optimization (CMA-ES) [72]. The goal of the covariance matrix adaptation is to estimate the reverse Hessian matrix, analogously to a quasi-Newton technique. Furthermore, to increase the utility of the DCMA-EA, the greedy selection method of DE is applied to improve individuals in the next generation [73]. CMA-ES uses a new differential perturbation structure, and the new population vector is shaped by the following equation:

$$X_{i,g} = m + \sigma\cdot N(0, c) = m + \ \sigma\cdot B\cdot D\cdot\ randn(dim)^T$$

where $randn(dim)^T$ is a group of random numbers taken from a normal distribution with zero mean and a standard deviation of 1 and has an element number equal to the dimensions of the function at hand. The parameter "m" and the evolution of "σ" determine the overall standard deviation.

By using a shared population, the new mutated vectors are produced as the target vectors as follows:

$$V_{i,g} = m\cdot(1 - P) + P\cdot X_{i,g} + F\cdot(x_{r_1^i,g} - \ x_{r_2^i,g}) + (1 - P)\cdot\ \sigma\cdot B\cdot d\cdot randn(dim)^T$$

where $x_{r_1^i,g}$ and $x_{r_2^i,g}$ are two vectors randomly selected from the population, m is the average of the present population, B is an orthonormal of eigenvectors, and D is the square root of the commensurate none negative eigenvalues. P is a control value which maintains the contribution of the average vector of the existing population and target ones as well. Both of the scale F and P are computed as follows:

$$F_i = 0.5 + 0.5 \cdot rand(0,1)$$

$$P = 0.5 \cdot (1 + \frac{iter}{iter - max})$$

3.3. Differential Evolution with Multiple Strategies

In this approach, four different mutation strategies and one crossover operator are used within a single algorithm framework, as proposed by Elsayed et al. [74]. The main objective is to adapt a mutation strategy by choosing one from a pool of allowable schemes. In fact, although this algorithm involves different mutation strategies with dissimilar features, the authors believe that these different strategies cannot yield suitable performance. Therefore, the performance of the mutation strategy is dependent on the progression of the evolution, which is based on the success of the search operators.

Therefore, the feasibility status and the fitness value factors are used to measure the enhancement in the infeasibility. If the problem becomes increasingly feasible, the improvement index is calculated as follows:

$$VI_{i,t} = \frac{\left| V_{i,t}^{best} - V_{i,t-1}^{best} \right|}{avg \cdot V_{i,t}} = I_{i,t}$$

where $V_{i,t}^{best}$ is the best individual at generation t and $avg \cdot V_{i,t}$ is the average of the violation.

3.4. Hybrid DE Algorithms

Hybridization is another way to increase convergence for optimization. Hybridized approaches balance global and local search techniques. Hybridization is the method of joining the advantages of two or more algorithms to produce one algorithm that is anticipated to generate better offspring [75]. Each approach has its strengths and weaknesses. Thus, by combining different approaches, performance is improved [76]. Hybridization can be implemented at four stages of interaction [76]. The first is at the individual stage for the search at examination level, which defines the performance of an individual in the population. The second is the population level, which appears as the dynamic range of a population. The third is the exterior level, which delivers communication with other methods. The fourth is the meta data level, in which a superior metaheuristic contains its strategies [77].

Each optimization technique has specific operators and procedures; for example, the DE algorithm consists of mutation, crossover and selection. In the hybridized technique, some operators can cooperate between two algorithms to exploit the complementary characteristics of different optimization strategies [78]. In fact, choosing a suitable combination of balanced algorithms is the key to achieving enhanced performance. Nevertheless, developing an effective hybrid algorithm is not easy because it requires proficiency in different areas of optimization. There are many types of problems for which a classic or modified differential evolution algorithm might fail to find a suitable solution [79]. Therefore, recently applied DE hybridization approaches have become widespread due to their ability to handle many real-world problems. Some of the benefits of DE hybridization have been previously discussed [18]. To enhance the performance of DE, such as the speed of convergence or the quality of DE, and to solve larger systems, DE must incorporate hybrid evolutionary methodologies [80]. In general, there are three types of hybridizations for evolutionary algorithms involving global optimization: hybridization with local search, hybridization with global optimization and hybridization involving both techniques [81]. In this section, we highlight and demonstrate several hybrid differential evolutionary algorithms reported in the literature.

3.5. Hybridization of DE with Other Evolution Algorithms

DE has been frequently hybridized with PSO because both algorithms implement simple difference processes to perturb the current population [82]. The variation between the current and the best individual is utilized both in the refresh population method of PSO and in the DE/current-to-best/1 mutation strategy.

The particle swarm optimization (PSO) method was proposed by J. Kennedy and R.C. Eberhart [83,84]. The technique shows perfect action compared with that of other evolutionary algorithms or metaheuristics. This approach mimics human cognition and has been applied to optimization problems. The goal is to apply a group of individuals called a swarm of particles [85]. The same notation used for DE is used for PSO; a vector is used as a solution for an optimization task t. At each loop t, a particle alteration index, affected by its velocity $v_i(t)$ via the equation $x_i(t) = x_i(t-1) + v_i(t)$. However, two equations control the updating of the velocity $v_i(t)$.

$$v_i(t) = v_i(t-1) + \rho_1 * (p_i - x_i(t-1))$$

g_{best} represent the whole population and lbest describes the subpopulation encompassing the particle. The g_{best} is practical for best results. Let p_g be the best results of the population; thus, social influence is mathematically expressed as $\rho_2 * (p_g - x_i(t-1))$. Therefore, updating the particles at each loop as follows:

$$v_i(t) = v_i(t-1) + \rho_1 * (p_i - x_i(t-1)) + \rho_2 * (p_g - x_i(t-1))$$

$$x_i(t) = x_i(t-1) + v_i(t)$$

where ρ_1 and ρ_2 are the control parameters.

PSO has several disadvantages, the most significant of which is its premature convergence. PSO consists of three components: previous velocities $v_i(t-1)$, present behavior $\rho_1 * (p_i - x_i(t-1))$, and social behavior $\rho_2 * (p_g - x_i(t-1))$.

Because PSO is built on these three components, it will not operate if any of those components has any issue; for example, a vector consisting of a bad solution will retard the optimal solution. However, DE does not carry the initial two features of PSO. The individual construct is based on a random walk algorithm in the search space, which then selects the optimal position index [86].

In PSO, the next position is based on the present optimal position p_i and by the particle's velocity v_i. In addition, the third feature of PSO could be inferred in DE as the RAND/BEST strategy. PSO refresh the velocity of a particle applying three expressions. In the proposed strategy, the particle velocities are updated by carrying the subtract of the index vectors of any two dissimilar particles arbitrarily selected from the swarm. Das et al. proposed PSO-DV (particle swarm with differentially perturbed velocity) [87]. In the proposed scheme, particle velocities are perturbed by a new term containing the weighted difference of the position vectors of any two dissimilar particles randomly selected from the swarm. This differential velocity term mimics the DE mutation [88]. PSO-DV applies the DE differential operator to update the velocity of PSO. Two vectors are chosen randomly from the population. Then, unlike in PSO, a particle is moved to a new position only if the new position produces a better fitness value. In PSO-DV, for each particle i in the swarm, two other separate particles j and k ($i \neq j \neq k$) are chosen randomly. The difference between their locations is calculated as a difference vector:

$$v_{id}(t+1) = \omega * v_{id}(t) + \beta * \delta_d + C_2\, \rho_2 * (p_{gd} - jx_{id}(t))$$

$$T_{ri} = x_i(t) + \beta * \delta_d + C_2 \rho_2 * (p_{gd} - jx_{id}(t))$$

where CR is the crossover rate, δ_d is the component of the subtract vector and β is a factor rate in the range [0, 1]. Hendtlass proposed the first combination of DE and PSO and called it SDEA as the individuals comply swarm principles [60]. DE is used to transfer the individuals to the promised region in a random fashion. Yu et al. proposed an adaptive hybrid algorithm based on PSO and

DE (HPSO-DE) with a composed population among PSO and DE [89]. The strategy incorporates the advantages of the two algorithms and maintains population diversity. Therefore, HPSO-DE has the ability to move to local optima [90]. Zhang et al. offered DEPSO, which apples a similar standard of updating PSO individuals via DE [91]. DEPSO performs well with numerical integer problems but is not efficient for small feasible space problems. Mutations are maintained by a DE operator on p_i, with a trail vector $T_i = p_i$ for the dth dimension:

$$if\ rand\ (\ )CR\ or\ d == k)\ then\ T_{id} = p_{gd} + \delta_{2,d}$$

where k is a random value within the domain [1, D], which includes that the mutation has at least one dimension. CR is a crossover constant, and δ_2, is the case of N = 2 for the general difference vector δ_N, which is defined as follows:

$$\delta_N = (\sum_{1}^{N} \Delta)/N$$

$$\Delta = P_A - P_B$$

where Δ is the difference vector and P_A and P_B are randomly chosen from the p-best set. Liu et al. offered a hybridization of PSO and DE in a pair of population scheme [92]. Three mutation strategies are borrowed from DE (DE/rand/1, DE/current_to_best/1DE/rand/2) are applied to refresh the former best solutions [93].

Trivedi et al. proposed a hybrid of DE and GA to resolve scheduling challenges [94]. GA operates on the binary element variables through the DE process to enhance the related power-related variables [95]. The advantage of a GA lies in its ability to discover a decent solution to a problem whenever the iterative approach is too time-consuming and the mathematical approach is unobtainable [96]. GA allows for the fast discovery of the solution. Although the genetic algorithm is not excessively complex, the parameters and implementation of the GA generally require a tremendous amount of tuning [97].

The advantage of DE is that, in general, it frequently shows better solutions than those yielded by GA and other evolutionary algorithms [98–100]. Furthermore, DE is easy to apply to a wide variety of problems regardless of noisy, multi-modal, multi-dimensional spaces, which typically make problems difficult to optimize. Although DE consists of two important parameters, Cr and F, those parameters do not require the same amount of tuning as those associated with other evolutionary algorithms [101]. Liao has proposed a hybridization of DE and a local search algorithm modeled after the harmony search (HS) algorithm to find the global optimum [102]. The main goal of this type of hybridization method is to advance the use of mixed discrete and real-valued-dimensional problems.

Boussaïd et al. proposed a hybridization of DE and biogeography based optimization (BBO) to deliver solutions through the optimal power distribution method in a wireless sensor network (WSN) [103,104].

Dulikravich et al. proposed a hybridized multi-objective, multi variable optimizer by combining non-dominated sorting differential evolution (NSDE) with the strength Pareto evolutionary algorithm (SPEA) and multi-objective particle swarm optimization (MOPSO) [105].

Guo and others have proposed a form of DE enhanced among self-adaptive parameters that depend on simulated annealing algorithms in the collection of DE; the classic selection technique is a greedy equation [106]. The greedy rule is easily trapped in a local optimum. However, a new selection technique based on simulated annealing is used in this algorithm. The approach is expressed as follows:

$$X_{i,G+1} = U_{i,G}\ if\ f(U_{i,G}) \leq f(X_{i,G})$$

$$X_{i,G+1} = U_{i,G}\ if\ f(U_{i,G}) f(X_{i,G})\ and\ \exp\left(-\frac{f(U_{i,G}) - f(X_{i,G})}{t_G}\right) rand\ (0,1)$$

$$X_{i,G+1} = X_{i,G}\ if\ f(U_{i,G})f(X_{i,G})\ and\ \exp\left(-\frac{f(U_{i,G}) - f(X_{i,G})}{t_G}\right) \leq rand\ (0,\ 1)$$

where t_G represents the Gth generation temperature. Pholdee and Bureerat offered a hybrid algorithm involving the trial vector method of DE called the real-coded population-based incremental learning (RCPBIL) algorithm [107]. The RPBIL can be extended to multi-objective optimization similarly to multi-objective PBIL using binary codes for which the population serves as a likelihood vector for single-objective problems [108]. When addressing multi-objective problems, more probability vectors are utilized to maintain population variety. Likewise, with the binary code of PBIL, the multi-objective style of the RPBIL uses numerous possibility matrix that appear for a real code population, where each probability matrix is called a tray [109].

A three-dimensional matrix, represents a group of probability trays with dimensions $n^*n_l^*n_T$, n_T is the number of trays required for each tray drive to be used to produce a real-code subpopulation, which has approximately N_P/N_T form results as its members.

An initial population is formed for the multi-starting search procedure with early likelihood trays. An initial Pareto archive is obtained, and non-dominated results are then designated to update the probability trays. Then, the centroid of the non-dominated solution set (R_G) is used to update a probability tray in the series, where the r_G of the set that has the lowest value of the first objective function is applied to update the first tray and so on.

The updating procedure for each tray can be improved by substituting X_{best} with r_G. Subsequently, a population yielding the updated trays is shaped. The Pareto archive is changed by substituting its members with non-dominated solutions saved from the mixture of the current population and the elements in the preceding archive. If the number of archive elements is larger than the constant archive size, the clustering method is initiated to eliminate non-dominated solutions from the archive. These steps are repeated until a stopping condition is fulfilled [110].

Neri et al. [111,112] proposed a compact DE hybridized with a memetic search to yield faster convergence [113]. The algorithm represents the population as a multi-dimensional Gaussian distribution and is called disturbed exploitation compact differential evolution (DEcDE) [114]. The DEcDE algorithm utilizes an evolutionary framework based on DE logic assisted by a shallow depth for processing the local search algorithm [114].

The output of the algorithm was introduced to create an MC model to gain high efficiency on a diverse set of problems, regardless of its limits, in terms of complexity and memory usage. At the start of the DEcDE algorithm, an $a(2.n)$ probability vector (PV) is produced.

$$PV^t = \left[\mu^t,\ \sigma^t\right]$$

where μ and σ are, respectively, the mean and standard deviation values for each design variable from a Gaussian probability distribution function (PDF) truncated within the interval $[-1, 1]$.

$$PDF = [\mu[i], \sigma[i]]$$

Zhan and Zhang proposed a differential evolution (DE) algorithm with a random walk (DE-RW) [115]. DE-RW is analogous to the classic DE algorithm, with a minor alteration in the crossover procedure that mixes the individual vector and the mutant vector to perform a random walk, forming the target vector as follows:

$$\begin{cases} v_{id}\ rand(0,1) > CR\ or\ d = k \\ rand(L_d, H_d),\ else\ if\ rand(0,1) > RW \\ x_{id} \quad otherwise \end{cases}$$

where L_d and H_d are the low and high search restrictions of the d_{ht} dimension and the parameter RW is used to control the effect of the random walk. The parameter RW is controlled as follows:

$$Rw = 0.1 - 0.099 \times g/G$$

where g and G are the current generation number and the maximum number of generations, respectively. A few notable DE algorithms are summarized in Table 2.

Table 2. Summary of different DE algorithms with variety of approaches.

Algorithm	Strategy	Note
Multi Population DE algorithm (MPDE) [116]	DE/best/1	MPDE created subpopulation in random manner from the main population and then the migration of the best vector from subpopulations to main population
Adaptive DE [117]	Six DE strategies and one strategy is randomly selected by a roulettewheel	Adaptively selects trial vector generation, scale factor "*F*" is 0.8 and it is constant for all strategies, also the crossover rate is constant as well that is 0.5
Self-Adapting Parameter Setting in Differential Evolution (jDE)	DE/rand/1/bin	jDE enhances the population size through the optimization procedure based on the developed DE parameters
Self-adaptive Mutation DE (SaMDE)	DE/rand/1, DE/best/1, DE/best/2andDE/current-to-rand/1	The strategy is chosen by a roulette wheel strategy. The scale factor is dynamic and chosen from a range (0.7; 1.0) after each generation.
Modified DE(MDE)with pbest crossover(MDE-pBX) [66]	DE/current-to-best/1, DE/current-to-gr_best/1 [gr indicate for group]	*F* and *Cr* directed by the information of their effective values that are capable of producing improved offspring
Modified DE algorithm (MDE)	DE/rand/1, DE/best/1	One of two strategies is chosen based on a probability.
DE with Self-Adaptive Mutation and Crossover (DESAMC)	Classic DE Strategy	Working to self-adapt the parameters values
Differential Covariance Matrix Adaptation Evolutionary Algorithm (CMA-ES)	New population vector is created using DE/rand/1/bin	Parameters are chosen randomly
Differential Evolution with Multiple Strategies	DE/best/1/bin,rand/1/bin, DE/best/1/exp, and DE/rand/1/exp	Parameters are chosen randomly
DE-PSO	Classic DE strategy + The two basic equations which govern the working of PSO	"DE-PSO" Hybrid differential evolution - Particle Swarm Optimization. The inclusion of PSO phase creates a perturbation in the population, which in turn helps in maintaining diversity of the population and producing a good optimal solution.
Hybrid of DE and GA (hGADE) [118]	Hybridized GA with only 2 classical DE variants	Randomly generated binary unit commitment matrices while the RPM of all the individuals in the initial population are generated
Hybridization of DE and Biogeography Based Optimization (BBO) [119]	Classical DE DE/rand/1/bin + classic BBO	The main operator of DE/BBO is the hybrid migration operator,

4. Conclusions

This paper presents a survey of DE and analyzes the modifications recently proposed in the literature to enhance the performance of DE. Various state-of–the-art differential evolution algorithms incorporating different strategies have been studied. DE performance is affected by the types of techniques applied. Enhanced DE algorithms are categorized into three groups: adaptive, self-adaptive and hybrid. However, as demonstrated in this article, seemingly minor modifications can greatly enhance the performance of DE. All the DE techniques and mechanisms described here offer partial improvements to the classic DE components to yield better performance. The selection of proper techniques in DE is associated with the mathematical characteristics of the problems addressed. Therefore, by properly adjusting the control parameters of different crossovers, similar levels of performance can be attained. This article offers an essential guide to differential evolution practitioners for acquiring optimum control parameters for DE. Thus, if information about the problem to be addressed is available, suitable optimal values should be adopted based on the basic guidelines and findings summarized in this paper. Researchers interested in DE can find valuable contributions in the literature about DE.

Author Contributions: Conceptualization of this paper survey which is carried to gain extensive ideas about DE and verities of techniques T.E created the structure and organization of the paper, reviewed and instituted the content in all sections and commented on the quantitative aspects of the DE algorithm. A.M. co-reviewed the paper. T.E. contributed to the final version of the manuscript. All authors approve of the final version of the manuscript.

Acknowledgments: The authors acknowledged the University of Bridgeport for the preparation of resources which are required to complete this research. The University of Bridgeport library contributes by arranging required books and researched papers in this work. The University of Bridgeport, CT USA also pledge us by funding the cost of publishing this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
2. Rao, S.S.; Rao, S.S. *Engineering Optimization: Theory and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
3. Price, K.; Storn, R.M.; Lampinen, J.A. *Differential Evolution: A Practical Approach to Global Optimization*; Springer Science & Business Media: Berlin, Germany, 2006.
4. Brownlee, J. *Clever Algorithms: Nature-Inspired Programming Recipes*; Jason Brownlee: Melbourne, Australia, 2011.
5. Zhang, J.; Sanderson, A.C. *Adaptive Differential Evolution*; Springer: Berlin/Heidelberg, Germany, 2009.
6. Feoktistov, V. *Differential Evolution*; Springer: Dordrecht, The Nerthelands, 2006.
7. Storn, R. On the usage of differential evolution for function optimization. In Proceedings of the NAFIPS, Biennial Conference of the North American Fuzzy Information Processing Society, Berkeley, CA, USA, 19–22 June 1996; pp. 519–523.
8. Price, K.V.; Storn, R.M.; Lampinen, J.A. The differential evolution algorithm. In *Differential Evolution: A Practical Approach to Global Optimization*; Price, K.V., Storn, R.M., Lampinen, J.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 37–134.
9. Salman, A.; Engelbrecht, A.P.; Omran, M.G. Empirical analysis of self-adaptive differential evolution. *Eur. J. Oper. Res.* **2007**, *183*, 785–804. [CrossRef]
10. Neri, F.; Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [CrossRef]
11. Onwubolu, G.C.; Davendra, D. *Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization*; Springer Science & Business Media: Berlin, Germany, 2009; Volume 175.
12. Peng, L.; Wang, Y. Differential evolution using uniform-quasi-opposition for initializing the population. *Inf. Technol. J.* **2010**, *9*, 1629–1634. [CrossRef]
13. Adeyemo, J.; Otieno, F. Differential evolution algorithm for solving multi-objective crop planning model. *Agric. Water Manag.* **2010**, *97*, 848–856. [CrossRef]
14. Chang, T.-T.; Chang, H.-C. Application of differential evolution to passive shunt harmonic filter planning. In Proceedings of the 8th International Conference on Harmonics and Quality of Power Proceedings, Athens, Greece, 14–16 October 1998; pp. 149–153.
15. Bergey, P.K.; Ragsdale, C. Modified differential evolution: A greedy random strategy for genetic recombination. *Omega* **2005**, *33*, 255–265. [CrossRef]
16. Fan, H.-Y.; Lampinen, J. A trigonometric mutation operation to differential evolution. *J. Glob. Optim.* **2003**, *27*, 105–129. [CrossRef]
17. Das, S.; Abraham, A.; Chakraborty, U.K.; Konar, A. Differential evolution using a neighborhood-based mutation operator. *IEEE Trans. Evol. Comput.* **2009**, *13*, 526–553. [CrossRef]
18. Qing, A. *Differential Evolution: Fundamentals and Applications in Electrical Engineering*; John Wiley & Sons: Singapore, 2009.
19. Lin, C.; Qing, A.; Feng, Q. A comparative study of crossover in differential evolution. *J. Heurist.* **2011**, *17*, 675–703. [CrossRef]
20. Guo, S.-M.; Yang, C.-C.; Hsu, P.-H.; Tsai, J.S.-H. Improving differential evolution with a successful-parent-selecting framework. *IEEE Trans. Evol. Comput.* **2015**, *19*, 717–730. [CrossRef]
21. Eiben, Á.E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **1999**, *3*, 124–141. [CrossRef]

22. Rozenberg, G.; Bäck, T.; Eiben, A.E.; Kok, J.N.; Spaink, H.P. (Eds.) *Natural Computing Series*; Springer: Berlin, Germany, 2006.
23. Wolpert, D.H.; Macready, W.G. *The Mathematics of Search*; Technical Report; Santa Fe Institute: Santa Fe, NM, USA, 1995.
24. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
25. Zaharie, D. On the explorative power of differential evolution. In Proceedings of the 3rd International Workshop on Symbolic and Numerical Algorithms on Scientific Computing, SYNASC-2001, Timişoara, Romania, 2–4 October 2001.
26. Liu, J. On setting the control parameter of the differential evolution method. In Proceedings of the 8th International Conference on Soft Computing (MENDEL 2002), Brno, Czech Republic, 5–7 June 2002; pp. 11–18.
27. Šmuc, T. Improving convergence properties of the differential evolution algorithm. In Proceedings of the MENDEL 2002-8th International Conference on Soft Computing, Brno, Czech Republic, 5–7 June 2002.
28. Yalcin, I.K.; Gokmen, M. Integrating differential evolution and condensation algorithms for license plate tracking. In Proceedings of the ICPR 2006, 18th International Conference on Pattern Recognition, Hong Kong, China, 20–24 August 2006; pp. 658–661.
29. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
30. Baíllo, Á.; Ventosa, M.; Rivier, M.; Ramos, A. Strategic bidding in a competitive electricity market: A decomposition approach. In Proceedings of the IEEE Porto Power Tech Proceedings, Porto, Portugal, 10–13 September 2001; Volume 1, p. 6.
31. Van Sickel, J.H.; Lee, K.Y.; Heo, J.S. Differential evolution and its applications to power plant control. In Proceedings of the ISAP, International Conference on Intelligent Systems Applications to Power Systems, Niigata, Japan, 5–8 November 2007; pp. 1–6.
32. Wang, X.; Cheng, H.; Huang, M. QoS multicast routing protocol oriented to cognitive network using competitive coevolutionary algorithm. *Expert Syst. Appl.* **2014**, *41*, 4513–4528. [CrossRef]
33. El Ela, A.A.; Abido, M.; Spea, S. Optimal power flow using differential evolution algorithm. *Electr. Power Syst. Res.* **2010**, *80*, 878–885. [CrossRef]
34. Goswami, J.C.; Mydur, R.; Wu, P. Application of differential evolution algorithm to model-based well log-data inversion. In Proceedings of the IEEE Antennas and Propagation Society International Symposium, San Antonio, TX, USA, 16–21 June 2002; pp. 318–321.
35. Su, C.-T.; Lee, C.-S. Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution. *IEEE Trans. Power Deliv.* **2003**, *18*, 1022–1027. [CrossRef]
36. Boughari, Y.; Ghazi, G.; Botez, R.M.; Theel, F. New Methodology for Optimal Flight Control Using Differential Evolution Algorithms Applied on the Cessna Citation X Business Aircraft—Part 1. Design and Optimization. *INCAS Bull.* **2017**, *9*, 31.
37. Price, K.V. Differential evolution vs. the functions of the 2/sup nd/ICEO. In Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, IN, USA, 13–16 April 1997; pp. 153–157.
38. Xue, F.; Sanderson, A.C.; Bonissone, P.P.; Graves, R.J. Fuzzy logic controlled multi-objective differential evolution. In Proceedings of the 14th IEEE International Conference on Fuzzy Systems, FUZZ '05, Reno, NV, USA, 25 May 2005; pp. 720–725.
39. Storn, R. *Differrential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*; Technical Report; International Computer Science Institute: Berkeley, CA, USA, 1995.
40. Mezura-Montes, E.; Velázquez-Reyes, J.; Coello Coello, C.A. A comparative study of differential evolution variants for global optimization. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, WA, USA, 8–12 July 2006; pp. 485–492.
41. Vesterstrom, J.; Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proceedings of the Congress on Evolutionary Computation, CEC2004, Portland, OR, USA, 19–23 June 2004; pp. 1980–1987.
42. Fister, I.; Mernik, M.; Brest, J. Hybridization of Evolutionary Algorithms. *arXiv* **2013**, arXiv:1301.0929.

43. Hu, C.; Yan, X. An immune self-adaptive differential evolution algorithm with application to estimate kinetic parameters for homogeneous mercury oxidation. *Chin. J. Chem. Eng.* **2009**, *17*, 232–240. [CrossRef]
44. Ilonen, J.; Kamarainen, J.-K.; Lampinen, J. Differential evolution training algorithm for feed-forward neural networks. *Neural Process. Lett.* **2003**, *17*, 93–105. [CrossRef]
45. Eiben, G.; Schut, M.C. New ways to calibrate evolutionary algorithms. In *Advances in Metaheuristics for Hard Optimization*; Siarry, P., Michalewicz, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 153–177.
46. Angeline, P.J. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*; IEEE Press: Piscataway, NJ, USA, 1995.
47. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin, Germany; London, UK, 2003; Volume 53.
48. Liu, J.; Lampinen, J.; Matousek, R.; Osmera, P. Adaptive parameter control of differential evolution. In Proceedings of the MENDEL, Brno, Czech Republic, 5–7 June 2002; pp. 19–26.
49. Abbass, H.A. The self-adaptive pareto differential evolution algorithm. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02, Honolulu, HI, USA, 12–17 May 2002; pp. 831–836.
50. Teo, J. Exploring dynamic self-adaptive populations in differential evolution. *Soft Comput.* **2006**, *10*, 673–686. [CrossRef]
51. Liu, J.; Lampinen, J. A fuzzy adaptive differential evolution algorithm. *Soft Comput.* **2005**, *9*, 448–462. [CrossRef]
52. Qin, A.K.; Suganthan, P.N. Self-adaptive differential evolution algorithm for numerical optimization. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; pp. 1785–1791.
53. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
54. Kaelo, P.; Ali, M. Differential evolution algorithms using hybrid mutation. *Comput. Optim. Appl.* **2007**, *37*, 231–246. [CrossRef]
55. Rocha, A.M.A.; Fernandes, E.M.d.G. On charge effects to the electromagnetism-like algorithm. In Proceedings of the 20th EURO Mini Conference: Continuous Optimization and Knowledge-Based Technologies, Neringa, Lithuania, 20–23 May 2008; pp. 198–203.
56. Wang, Y.; Cai, Z.; Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [CrossRef]
57. Zaharie, D. Control of population diversity and adaptation in differential evolution algorithms In Proceedings of the Mendel, 9th International Conference on Soft Computing, Brno, Czech Republic, 26–28 June 2003.
58. Kumar, P.; Pant, M. A self adaptive differential evolution algorithm for global optimization. In Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing, Chennai, India, 16–18 December 2010; pp. 103–110.
59. Brest, J.; Bošković, B.; Greiner, S.; Žumer, V.; Maučec, M.S. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Comput.* **2007**, *11*, 617–629. [CrossRef]
60. Hendtlass, T. A combined swarm differential evolution algorithm for optimization problems. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Budapest, Hungary, 4–7 June 2001; pp. 11–18.
61. Yang, Z.; Tang, K.; Yao, X. Differential evolution for high-dimensional function optimization. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 25–28 September 2007; pp. 3523–3530.
62. Ling, M.-X.; Wang, F.-Y.; Ding, X.; Hu, Y.-H.; Zhou, J.-B.; Zartman, R.E.; Yang, X.-Y.; Sun, W. Cretaceous ridge subduction along the lower Yangtze River belt, eastern China. *Econ. Geol.* **2009**, *104*, 303–321. [CrossRef]
63. Babu, B.; Angira, R. Modified differential evolution (MDE) for optimization of non-linear chemical processes. *Comput. Chem. Eng.* **2006**, *30*, 989–1002. [CrossRef]
64. Sayah, S.; Zehar, K. Modified differential evolution algorithm for optimal power flow with non-smooth cost functions. *Energy Convers. Manag.* **2008**, *49*, 3036–3042. [CrossRef]
65. Lakshminarasimman, L.; Subramanian, S. Short-term scheduling of hydrothermal power system with cascaded reservoirs by using modified differential evolution. *IEE Proc.-Gener. Transm. Distrib.* **2006**, *153*, 693–700. [CrossRef]

66. Islam, S.M.; Das, S.; Ghosh, S.; Roy, S.; Suganthan, P.N. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2012**, *42*, 482–500. [CrossRef] [PubMed]
67. Cai, Y.; Wang, J. Differential evolution with neighborhood and direction information for numerical optimization. *IEEE Trans. Cybern.* **2013**, *43*, 2202–2215. [CrossRef] [PubMed]
68. Alguliev, R.M.; Aliguliyev, R.M.; Isazade, N.R. DESAMC+ DocSum: Differential evolution with self-adaptive mutation and crossover parameters for multi-document summarization. *Knowl.-Based Syst.* **2012**, *36*, 21–38. [CrossRef]
69. Selamat, A.; Nguyen, N.T.; Haron, H. Intelligent Information and Database Systems. In Proceedings of the 5th Asian Conference, ACIIDS 2013, Kuala Lumpur, Malaysia, 18–20 March 2013; Springer: Berlin/Heidelberg, German, 2013; Volume 7803.
70. Zhang, J.; Sanderson, A.C. JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 25–28 September 2007; pp. 2251–2258.
71. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013; pp. 71–78.
72. Ghosh, S.; Das, S.; Roy, S.; Islam, S.M.; Suganthan, P.N. A differential covariance matrix adaptation evolutionary algorithm for real parameter optimization. *Inf. Sci.* **2012**, *182*, 199–219. [CrossRef]
73. Ghosh, S.; Roy, S.; Islam, S.M.; Das, S.; Suganthan, P.N. A differential covariance matrix adaptation evolutionary algorithm for global optimization. In Proceedings of the 2011 IEEE Symposium on Differential Evolution (SDE), Paris, France, 11–15 April 2011; pp. 1–8.
74. Elsayed, S.M.; Sarker, R.A.; Essam, D.L. An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Trans. Ind. Inform.* **2013**, *9*, 89–99. [CrossRef]
75. Lichtblau, D. Relative position indexing approach. In *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 81–120.
76. Blum, C.; Puchinger, J.; Raidl, G.R.; Roli, A. Hybrid metaheuristics in combinatorial optimization: A survey. *Appl. Soft Comput.* **2011**, *11*, 4135–4151. [CrossRef]
77. Dragoi, E.-N.; Dafinescu, V. Parameter control and hybridization techniques in differential evolution: A survey. *Artif. Intell. Rev.* **2016**, *45*, 447–470. [CrossRef]
78. Goldberg, D.E.; Holland, J.H. Genetic algorithms and machine learning. *Mach. Learn.* **1988**, *3*, 95–99. [CrossRef]
79. Vas, P. *Artificial-Intelligence-Based Electrical Machines and Drives: Application of Fuzzy, Neural, Fuzzy-Neural, and Genetic-Algorithm-Based Techniques*; Oxford University Press: Oxford, UK, 1999; Volume 45.
80. Panigrahi, B.K.; Suganthan, P.N.; Das, S.; Dash, S.S. Swarm, Evolutionary, and Memetic Computing. In Proceedings of the Third International Conference SEMCCO, Chennai, India, 16–18 December 2010.
81. Nwankwor, E.; Nagar, A.K.; Reid, D. Hybrid differential evolution and particle swarm optimization for optimal well placement. *Comput. Geosci.* **2013**, *17*, 249–268. [CrossRef]
82. Vitaliy, F. *Differential Evolution—In Search of Solutions*; Springer: New York, NY, USA, 2006.
83. Fister, I.; Fister, I., Jr. *Adaptation and Hybridization in Computational Intelligence*; Springer: Cham, Switzerland, 2015; Volume 18.
84. Thangaraj, R.; Pant, M.; Abraham, A.; Bouvry, P. Particle swarm optimization: Hybridization perspectives and experimental illustrations. *Appl. Math. Comput.* **2011**, *217*, 5208–5226. [CrossRef]
85. Eberhart, R.; Kennedy, J. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Network, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
86. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the Congress on Evolutionary Computation, CEC '99, Washington, DC, USA, 6–9 July 1999; pp. 1945–1950.
87. Trelea, I.C. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Inf. Process. Lett.* **2003**, *85*, 317–325. [CrossRef]
88. Kennedy, J.; Mendes, R. Population structure and particle swarm performance. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC '02, Honolulu, HI, USA, 12–17 May 2002; pp. 1671–1676.

89. He, Z.; Wei, C.; Yang, L.; Gao, X.; Yao, S.; Eberhart, R.C.; Shi, Y. Extracting rules from fuzzy neural network by particle swarm optimisation. In Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 74–77.
90. Das, S.; Abraham, A.; Konar, A. Particle swarm optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives. In *Advances of Computational Intelligence in Industrial Systems*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 1–38.
91. Van Sickel, J.H.; Lee, K.Y.; Heo, J.S. Differential evolution and its applications to power plant control. In Proceedings of the International Conference on Intelligent Systems Applications to Power Systems (ISAP 2007), Niigata, Japan, 5–8 November 2007; pp. 1–6.
92. Yu, X.; Cao, J.; Shan, H.; Zhu, L.; Guo, J. An adaptive hybrid algorithm based on particle swarm optimization and differential evolution for global optimization. *Sci. World J.* **2014**, *2014*, 215472. [CrossRef] [PubMed]
93. Zhang, W.-J.; Xie, X.-F. DEPSO: Hybrid particle swarm with differential evolution operator. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Washington, DC, USA, 8 October 2003; pp. 3816–3821.
94. Liu, B. Uncertain risk analysis and uncertain reliability analysis. *J. Uncertain Syst.* **2010**, *4*, 163–170.
95. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [CrossRef]
96. Hästbacka, J.; de la Chapelle, A.; Mahtani, M.M.; Clines, G.; Reeve-Daly, M.P.; Daly, M.; Hamilton, B.A.; Kusumi, K.; Trivedi, B.; Weaver, A. The diastrophic dysplasia gene encodes a novel sulfate transporter: Positional cloning by fine-structure linkage disequilibrium mapping. *Cell* **1994**, *78*, 1073–1087. [CrossRef]
97. Das, S.; Abraham, A.; Konar, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2008**, *38*, 218–237. [CrossRef]
98. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison: Reading, MA, USA, 1989.
99. Mezura-Montes, E. Nature-Inspired Algorithms Evolutionary and Swarm Intelligence Approaches. In Proceedings of the 7th Mexican Internatioal Conference of Arifical Intelligence "MICAI", Instituto Tecnol ogico de Monterrey, Monterrey, Mexico, 27–31 October 2008.
100. Xu, X.; Li, Y. Comparison between particle swarm optimization, differential evolution and multi-parents crossover. In Proceedings of the International Conference on Computational Intelligence and Security, Harbin, China, 15–19 December 2007; pp. 124–127.
101. Codreanu, I. A parallel between differential evolution and genetic algorithms with exemplification in a microfluidics optimization problem. In Proceedings of the International Semiconductor Conference (CAS 2005), Sinaia, Romania, 3–5 October 2005; pp. 421–424.
102. Sentinella, M.R. Comparison and integrated use of differential evolution and genetic algorithms for space trajectory optimisation. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, 25–28 September 2007; pp. 973–978.
103. Hegerty, B.; Hung, C.-C.; Kasprak, K. A comparative study on differential evolution and genetic algorithms for some combinatorial problems. In Proceedings of the 8th Mexican International Conference on Artificial Intelligence, Seattle, WA, USA, 8–12 July 2006.
104. Liao, T.W. Two hybrid differential evolution algorithms for engineering design optimization. *Appl. Soft Comput.* **2010**, *10*, 1188–1199. [CrossRef]
105. Boussaïd, I.; Chatterjee, A.; Siarry, P.; Ahmed-Nacer, M. Two-stage update biogeography-based optimization using differential evolution algorithm (DBBO). *Comput. Oper. Res.* **2011**, *38*, 1188–1198. [CrossRef]
106. Boussaïd, I.; Chatterjee, A.; Siarry, P.; Ahmed-Nacer, M. Hybridizing biogeography-based optimization with differential evolution for optimal power allocation in wireless sensor networks. *IEEE Trans. Veh. Technol.* **2011**, *60*, 2347–2353. [CrossRef]
107. Moral, R.; Sahoo, D.; Dulikravich, G. Multi-objective hybrid evolutionary optimization with automatic switching. In Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, VA, USA, 6–8 September 2006.
108. Guo, H.; Li, Y.; Li, J.; Sun, H.; Wang, D.; Chen, X. Differential evolution improved with self-adaptive control parameters based on simulated annealing. *Swarm Evol. Comput.* **2014**, *19*, 52–67. [CrossRef]

109. Pholdee, N.; Bureerat, S.; Yıldız, A.R. Hybrid real-code population-based incremental learning and differential evolution for many-objective optimisation of an automotive floor-frame. *Int. J. Veh. Des.* **2017**, *73*, 20–53. [CrossRef]
110. Pholdee, N.; Bureerat, S. Hybridisation of real-code population-based incremental learning and differential evolution for multiobjective design of trusses. *Inf. Sci.* **2013**, *223*, 136–152. [CrossRef]
111. Pholdee, N.; Bureerat, S. Hybrid real-code population-based incremental learning and approximate gradients for multi-objective truss design. *Eng. Optim.* **2014**, *46*, 1032–1051. [CrossRef]
112. Bureerat, S.; Pholdee, N.; Park, W.-W.; Kim, D.-K. An Improved Teaching-Learning Based Optimization for Optimization of Flatness of a Strip During a Coiling Process. In Proceedings of the International Workshop on Multi-disciplinary Trends in Artificial Intelligence, Chiang Mai, Thailand, 7–9 December 2016; pp. 12–23.
113. Neri, F.; Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm Evol. Comput.* **2012**, *2*, 1–14. [CrossRef]
114. Zou, D.; Wu, J.; Gao, L.; Li, S. A modified differential evolution algorithm for unconstrained optimization problems. *Neurocomputing* **2013**, *120*, 469–481. [CrossRef]
115. Zhan, Z.-H.; Zhang, J. Enhance differential evolution with random walk. In Proceedings of the ACM 14th Annual Conference Companion on Genetic and Evolutionary Computation, Philadelphia, PA, USA, 7–11 July 2012; pp. 1513–1514.
116. Yu, W.-J.; Zhang, J. Multi-population differential evolution with adaptive parameter control for global optimization. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12–16 July 2011; pp. 1093–1098.
117. Bujok, P.; Tvrdik, J.; Polakova, R. Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 2253–2258.
118. Trivedi, A.; Srinivasan, D.; Biswas, S.; Reindl, T. A genetic algorithm–differential evolution based hybrid framework: Case study on unit commitment scheduling problem. *Inf. Sci.* **2016**, *354*, 275–300. [CrossRef]
119. Pant, M.; Thangaraj, R.; Grosan, C.; Abraham, A. Hybrid differential evolution-particle swarm optimization algorithm for solving global optimization problems. In Proceedings of the ICDIM 2008, Third International Conference on Digital Information Management, London, UK, 13–16 November 2008; pp. 18–24.